A Bare-Hand Gesture Interaction System for Virtual Environments

Benjamín Hernández[1] and Alejandro Flores[2]
[1]*Computer Sciences Department, Barcelona Supercomputing Center, Barcelona, Spain*
[2]*R&D Department, Espora Estudio, Mexico City, Mexico*
benjamin.hernandez@bsc.es, alejandro@esporaestudio.com

Keywords: Hand Gesture Recognition, Real Time, Hand Based Interaction, Virtual Environments, Depth Sensor.

Abstract: Hand-based gestures provide direct mappings of user actions to 3D UI tasks; they are becoming a more attractive interaction alternative than keyboards, mice, controllers, among others. In this paper, we present a fast algorithm for hand gesture recognition for interaction in virtual environments. The method applies mathematical morphology operators (binarization and dilation) to acquire a clean segmented hand image from a depth data stream on which a curvature-metric and K-means algorithm is applied to detect the fingertips, then using fingertip and palm positions together with anthropomorphic metrics and a rule based system we perform gesture recognition. In addition, the intermittence gesture spotting problem is reduced using a digital integrator. Finally, a set of virtual environments were designed to demonstrate the performance, reliability and feasibility of our method.

1 INTRODUCTION

As Sturman and Zeltzer stated, our primary physical connection to the world is through our hands (Sturman and Zeltzer, 1994). Hand gestures enhance or even substitute oral communication between people. Using our hands as an input device provides a more direct mapping of our actions to interaction tasks; therefore, we can apply all our intellect to those tasks. In this sense, commodity depth cameras have enabled simpler ways of interaction known as natural user interface (NUI). This interface is primarily based on tracking and recognition of body motions and gestures; however, hand based interaction remains an open problem.

Hand based interaction has three main challenges: hand detection, hand tracking and gesture recognition. Hand detection and tracking, intimately related, consists in capturing the hand features (i.e. the palm, fingers and/or fingertips) and position over time while hand gesture recognition generates semantical information from tracking. Seminal works based on the use of glove technology (DeFanti and Sandin, 1977; Zimmerman et al., 1987) offered a solution to the hand tracking problem. However, wearing a glove may be intrusive, in particular when users need to manipulate tools.

Vision based techniques using video cameras are an unobtrusive alternative to glove technology; they focus mainly on hand detection and tracking using sophisticated algorithms to overcome the limited optical features of these devices. Commodity depth cameras provide distance range information that can be used to design computationally inexpensive algorithms.

Lightweight and robust algorithms for hand based interaction are crucial for virtual environments. Computer generated environments make intensive use of complex algorithms for collision detection, lighting, particle systems simulation, visualization, among others. Rendering should maintain a steady refresh rate of 33 ms while interaction response should take $0.1s$ for the user to feel that the system is reacting instantaneously.

Contribution. Enabling computers to understand the user's actions is the main problem that hand detection, tracking and gesture recognition address. In addition, virtual environments require efficient approaches to this problem. Our contribution is a low-latency and robust algorithm for bare-hand gesture recognition and real time interaction in virtual environments. The proposed algorithm makes use of a depth camera (Microsoft Kinect) to generate semantic information about the user hand gestures through five stages: binarization, dilation, fingertip detection, gesture recognition and filtering. We will show that our algorithm performs faster (7.62 ms per frame) and it is robust enough (86 % avg. gesture recognition rate) when compared with current approaches using depth

cameras. In addition, a set of virtual environments were designed as a testbed for our approach.

2 RELATED WORK

Different hand tracking and gesture recognition surveys (Sturman and Zeltzer, 1994; Garg et al., 2009; Zabulis et al., 2009) have classified hand tracking and recognition technologies based on particular features. (Sturman and Zeltzer, 1994) classified hand tracking technology in two main groups: position tracking (categorized in marker systems, silhouette analysis, optical, magnetic and acoustic tracking systems) and glove based input. (Garg et al., 2009) made emphasis on "data-glove" and vision based approaches while (Zabulis et al., 2009) presented an exhaustive review of vision based techniques using ordinary video cameras.

Seminal works on hand detection and tracking based on depth information made use of two cameras to recover a 27 degrees of freedom (DOF) of the hand (Rehg and Kanade, 1993) or used multiple cameras effectively in the context of 3D scene creation (Utsumi, 1997). A recent work (Wang et al., 2011) proposed a bimanual hand tracking system that provides a 6-DOF control for 3D assembly. Multiple camera systems require stereo matching, which imposes performance constraints. However by the introduction of depth cameras this process is simplified. In addition, it conveys two advantages: first, 3D data acquisition is in large part insensitive to illumination changes as long as the environment does not contain light components that interferes with the 3D sensor's active illumination (e.g. the infrared sunlight's component); second, segmentation is done with ease using the captured distance range information.

The ZCam, a time-of-flight camera designed by 3DV Systems (Iddan and Yahav, 2001) and one of the first depth-image capture device, was used in (Liu and Fujimura, 2004) for hand gesture recognition. They defined a gesture space based on hand shape, location and motion information. Hand shape analysis was done using a dataset of hand patterns images; to match a hand shape they used the chamfer distance between the input image and this dataset. They also performed hand trajectory analysis using the minimum square error between the user's performed trajectory and templates in a curve dataset. An usability problem with this technique arises when new gestures are needed, a special recording session is required to capture their corresponding pattern images; in this sense we opted for a specification of simple rules, based on the detected fingers' direction, fingertips and palm positions. Another potential problem can arise when dataset increases its size since more matching tests between the input gestures and stored hand patterns images need to be performed resulting in performance degradation, an important feature that must be preserved in real-time virtual environment interaction.

On the other hand, methods that use Kinect such as (Iason Oikonomidis and Argyros, 2011; Oikonomidis et al., 2011; Raheja et al., 2011; Frati and Prattichizzo, 2011) are mainly focused on hand tracking. In (Iason Oikonomidis and Argyros, 2011; Oikonomidis et al., 2011) a 26-DOF hand tracking system was described using particle swarm optimization (PSO) to minimize the discrepancy between the appearance and 3D structure of hypothesized instances of a hand model and actual hand observations. Despite the fact that these approaches are implemented in a last generation GPU, they can unlikely be used for real-time interaction in virtual environments since performance is about $66ms$ (Iason Oikonomidis and Argyros, 2011) and $50ms$ (Oikonomidis et al., 2011). In (Raheja et al., 2011) a method for palm and finger tracking was presented that extensively uses an already existent OpenNI module based on Bayesian Object Localization. In (Frati and Prattichizzo, 2011) was proposed the use of a haptic device, a Kinect sensor and a heuristic hand tracker to simulate force feedback: according to the authors haptic devices worn by the user does not affect tracking; however, wearing a device to make tracking more robust can be avoided. (Liang et al., 2012) proposed a method for fingertip and palm tracking emphasising cases where correct fingers' position detection can be problematic such as side-by-side fingers, bending fingers or nearby fingertips. They implemented a restricted geodesic shortest path metric and a particle filter to track the fingertips correctly while palm tracking was performed with a Kalman filter.

Kinect has also been used for hand gesture recognition. For example, in (Ren et al., 2011) is presented a method for correct finger detection in cases where fingers are too close to each other. Ren et al. used a modified version of the Earth Mover's Distance (Rubner et al., 2000), called Finger-Earth Mover's Distance, to penalize unmatched fingers in order to improve gesture detection. In (Suau et al., 2011) a setup which uses Kinect and SR4000 time-of-flight camera is presented. In this work, the position of the user's head is estimated with a depth-based template matching and an adaptive search zone. Then, hands are detected in a bounding box attached to a head's estimated position so that the user may move freely in the scene. This method can perform basic gesture

recognition based on the overall shape of the hands. More complex gestures based on fingers are not feasible since they down sample the captured depth image to achieve real-time performance.

Hand gesture interaction in virtual environments requires constant evaluation of the motion and gestures performed by the user, graphics and other multi modal elements must be synchronized. Simulating and visualizing large environments efficiently also require a significant use of assets that should be minimized. Our method is designed to meet these requirements; we will show that its balance between performance and accuracy makes it suitable for real-time interaction. In addition, our method does not require extra training sessions or additional training datasets, maintaining memory requirements low.

3 METHOD

The input to our method is a video stream containing depth information of the user interacting with our system. We begin by detecting the user's raised hand by using mathematical morphology operations (Sec. 3.1). Once the hand has been segmented, and all remaining depth information has been discarded, we detect the visible fingertips based on curvature measures and k-mean clustering (Sec. 3.2). Gesture detection uses fingertip information, and other anthropomorphic metrics to recognize different gestures. Once a given gesture has been recognized, the algorithm records the general trajectory of such gesture to get additional semantic information (Sec. 3.3). Finally, we apply a digital integrator (Sec. 3.4) to reduce the intermittent gesture spotting problem occurring due to the dynamically variation in shape and for a single gesture.

3.1 Binarization and Dilation

Common camera systems usually perform hand detection based on color, shape, pixel appearance or texture segmentation (Zabulis et al., 2009). Problems in color segmentation are produced by background objects and changing lighting conditions. Pixel appearance or texture segmentation make use of sophisticated machine learning techniques requiring a considerable amount of training data, learning time and computational power. In contrast, the use of a 3D sensor simplifies hand detection to an image segmentation problem based on depth. We solve this problem by applying mathematical morphology operators such as binarization and dilation on the captured depth image, in addition, these operations are implemented in GPU resulting in a low latency hand detection stage.

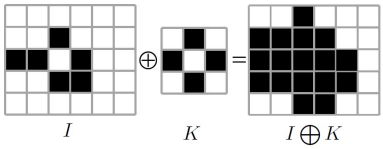
Figure 1: Binarization and dilation. Left: captured depth image. Right: binarized image.

Binarization consists in defining a threshold value and comparing every depth image's pixel with this threshold. If a given pixel value is below such threshold, the pixel is set to a minimal value; otherwise it is set to a maximum value. Binarization was performed on a 640×480 depth data stream, with minimal, maximum and threshold values initialized to 0, 255 and 18 pixels respectively. Figure 1 shows a captured depth image (left) and the resulting image after applying the binarization operator (right).

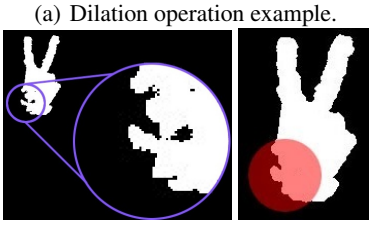
After binarization recurrent artifacts appeared in hand contour due the limited Kinect resolution (fig. 2b), which are a significant noise source in subsequent stages. These artifacts are reduced by dilation operation: let I be a binary image and K a structuring element or kernel. The dilation of I and K is defined in equation 1.

$$I \bigoplus K = \bigcup_{k \in K} I_k \tag{1}$$

This operation can be understood as the locus of the points covered by K when the center of K moves inside I, in other words, it is similar to the "convolution" of K in I but it changes or not the current pixel's value according to its neighbor pixels' value (fig. 2a); dilation can also be applied several times to improve results. Figure 2c shows the final hand detection's result after binarization and dilation operations.



(a) Dilation operation example.



(b) (c)

Figure 2: (b) High level curvature artifacts after binarization stage. (c) Resultant image after dilation stage.

3.2 Fingertip Detection

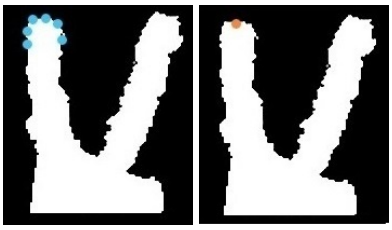
Fingertip detection is performed in two steps: first, we generate a reduced list of candidate points gathered from the finger zones; second, fingertip detection is performed by applying k-means clustering on this list. The list of candidate points, or candidate fingertips, is calculated by searching high curvature zones given the curvature measure (Argyros and Lourakis, 2006) shown in equation 2a.

$$C(P_j) = \frac{\vec{P_{j-k}P_j} \cdot \vec{P_jP_{j+k}}}{\left\|\vec{P_{j-k}P_j}\right\| \left\|\vec{P_jP_{j+k}}\right\|} \quad (2a)$$

$$\alpha = \arccos(C(P_j)) \quad (2b)$$

Where P_{j-k}, P_j and P_{j+k} are a set of successive points on the hand contour and $C(P_j)$ is the curvature measure of P_j.

We reduce the list of candidate points to guarantee k-means clustering computes only meaningful points. After calculating a curvature measure, $C(P_j)$, equation 2b is evaluated. If α is less than 60 degrees, $C(P_j)$ is added to the list, otherwise is discarded. After the list is completed (fig. 3a), k-means clustering is performed to obtain the correct fingertip (fig. 3b).



(a) (b)

Figure 3: (a) Equations 2a and 2b are used to obtain a reduced list of points at the finger zones. (b) Calculating K-means clustering results in correct fingertip detection.

Finally, finger labeling is performed by calculating the distance from the palm's center (obtained from the hand's bounding box) to each fingertip. Based on anthropomorphic metrics (Greiner, 1991), in particular on the relation of these distances, we identify each fingertip, i.e. the middle finger will usually be the longest finger, the remaining longest one is the index and so on (fig. 4).

3.3 Gesture Detection

Our system recognizes hand gestures and hand's trajectory gestures. Hand gestures are detected using a state based system; a state, denoting a gesture, is reached when specific interval values are tracked. Let

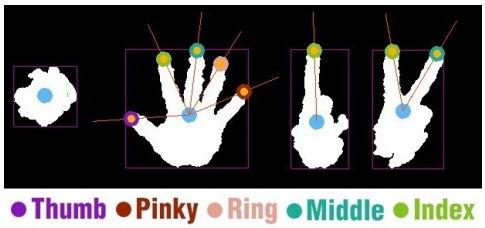


Figure 4: Fingertip labeling (color coded) in different situations.

S be a reached state, a_h a pair of values denoting minimum and maximum hand's area in pixels, n_f the number of fingertips found and d_f the directions from the palm to each fingertip, then a state is defined by $S =< a_h, n_f, d_f >$.

Figure 5 shows a set of hand gestures we defined for test purposes. Hand closed gesture is defined by $S =< [7240, 7281], 0, 0 >$, hand open gesture by $S =< maxArea, 5, any >$, pointing is detected when $S =< any, 1, upwards >$ and V-sign gesture is defined by $S =< any, 2, upwards >$.

Each state is evaluated as follows: first, the algorithm verifies if fingertips were detected, if not, it assumes the hand is closed then hand closed gesture detection starts based on its defined state. Second, if any fingertip has been detected, the algorithm begins the evaluation depending on the number of fingertips found, e.g. the V-sign gesture detection begins when two fingertips are found then if these two fingertips were previously labeled as index and middle (sec. 3.2) and both are pointing upwards the V-Sign has been detected.

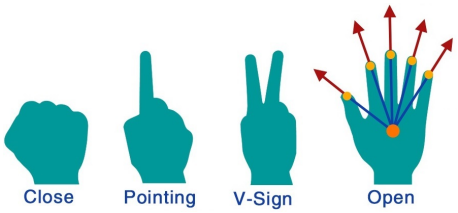
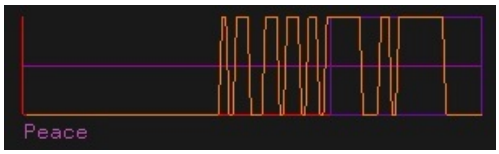


Figure 5: Test set of hand gestures.

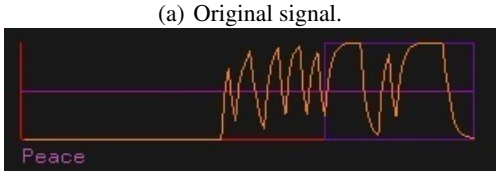
To detect the hand's trajectory gestures, we prefer simplicity and performance over sophistication. Some approaches (Liu and Fujimura, 2004; Plamondon and Srihari, 2000) make extensive use of machine learning techniques such as Hidden Markov Models, neural networks or featured based statistical classifiers to obtain semantical information related to a performed motion. These techniques usually requires training sessions and data that can be avoided using a simpler approach proposed by Wobbrock et al. based on a modified version of adjustment by least squares, called $1 Recognizer (Wobbrock et al., 2007).

3.4 Filtering

A typical problem in gesture recognition systems is intermittent gesture spotting, which is the result of the dynamic variation in shape and duration of a same gesture. We face this issue as a digital signal processing problem by generating and filtering a binary signal over time. The signal is generated as follows: when a gesture is detected the signal's value is set to one otherwise is set to zero (figure 6a). Intermittent gesture spotting expresses itself as high frequency variations in the signal. These variations are reduced using a digital integrator or a low-pass digital filter. A digital integrator can be implemented using different approaches such as finite-impulse-response (FIR) or infinite-impulse-response (IIR). In our case, we use a FIR integrator because feedback is not present, thus errors are discarded in subsequent iterations making them stable. Equation 3 models the FIR integrator implemented in our system.



(a) Original signal.



(b) Filtered signal.

Figure 6: "Peace" gesture spotting signal.

$$x'[n] = b_0 x[n] + b_1 x[n-1] + ... + b_N x[n-N] \quad (3)$$

Where $x[n]$ is the input signal, $x'[n]$ is the filtered signal, b_i are the filter coefficients or tap weights and N is the filter order. Figure 6b, shows the results of applying this filter to the "Peace" gesture spotting signal in a sequence of 127 samples. Notice the signal starts to decay slowly which reduces considerably the gesture spotting intermittence. As a result a more robust and stable gesture recognition method is obtained. Further results with N set to 24, 42 and 100 are shown in table 1 and explained in section 4.

4 EXPERIMENTAL RESULTS

Our recognizer was partially implemented in CUDA (Sec. 3.1) and C++ (Sec. 3.2, 3.3, 3.4) using OpenNI to capture Kinect's depth data stream and Open Sound Control protocol to send gesture detection state messages from the gesture detection program to the virtual environment applications, designed in Unity Engine. The virtual environment applications where designed to test common 3D UI tasks such as navigation, selection and manipulation (fig. 7).

1. ***Virtual woman dissection.*** The objective of this application was to test the manipulation of a 3D model through a 2D GUI. Navigation and selection of sliders are performed with the open hand gesture and closed hand gesture respectively. Manipulation (rotation, scaling and translation) is performed according to the selected slider by vertically moving the closed hand gesture. An additional GUI control changes the rendering mode to visualize different body systems (fig. 7 row 1).
2. ***Medieval Town.*** In this application, the user can navigate in a 3D environment. The user has to perform the close hand gesture to move the camera forwards. In addition, the user can see the surroundings using the pointing gesture once and moving left or right the pointing finger. The user can also throw knives by performing the V-Sign gesture (fig. 7 row 2).
3. ***Metaphors.*** In this environment, the user can interact with objects through similar gestures used in real life. First, the user has to open a door by moving his hand with the open gesture over the handle, after opening the door, the camera moves in front of a staple, the user touches the staple with his hand closed to staple. Finally, the camera moves in front of a turntable, the user has to open his hand over for scratching (fig. 7 row 3).
4. ***Trivia game.*** This application was designed to test hand trajectory gestures. The users have to answer a trivia by tracing in the air the numbers 1, 2 and 3 (fig. 7 row 4 left) to answer a question (fig. 7 row 4 right).

We designed two experiments to verify the performance, reliability and feasibility of our solution. Performance refers how fast is our algorithm, reliability to the level of success in gesture detection, and feasibility to how well hand gestures adapt to 3D UI tasks.

The first experiment consisted in measuring the performance and reliability of our recognizer. CUDA and C++ offer methods (cudaEventCreate(), cudaEventRecord(), cudaEventSynchronize() and cudaEventElapsedTime() were used in CUDA and clock() was used in C++) to measure the execution time of each stage of our algorithm. In a desktop computer with a i7-2600K CPU @ 3.40 GHz, GeForce GTX 550 TI GPU and 8 GB RAM, binarization and

Table 2: Comparison of our Algorithm and other Kinect based methods. Most of the technique use a similar hardware configuration as ours.

Method	Tracking	Gesture Recog.	Performance (ms)	Accuracy	Application
(Iason Oikonomidis and Argyros, 2011)	Yes	No	66.6	74% the estimated pose deviated 4cm or less	Not tested
(Oikonomidis et al., 2011)	Yes	No	50	60 (using one camera) 3.02 (using 8 cameras) Lower is better	Grasping real objects
(Frati and Prattichizzo, 2011)	Yes	No	350 to 30 after 70 iterations	Not reported	Haptic feedback grasping of a virtual cube.
(Raheja et al., 2011)	Yes	No	Not reported	near 100 % (fingers) near 90 % (palm)	Not tested.
(Liang et al., 2012)	Yes	No	Not reported	Different values for each finger e.g. 2.51 / 1.34 thumb (Lower is better)	Grasping of a virtual sphere
(Ren et al., 2011)	No	Yes	500 / 4000	90.6 % / 93.9% (Higher is better)	2D UI for Arithmetics; Rock-paper-scissors game.
(Suau et al., 2011)	Yes	Yes	14.7	*	2D Navigation
Our approach	Yes	Yes	7.62	Table1	Interaction in VR.

Table 1: Reliability results.

Gesture	# of frames	# Positive detection	Accuracy (%)
	N = 24	avg = 80.74 %	
Pointing	267	155	58.05
V-Sign	655	609	92.98
Open	235	201	85.53
Closed	500	432	86.40
	N = 42	avg = 88.07 %	
Pointing	407	374	91.89
V-Sign	219	179	81.74
Open	262	227	86.64
Closed	538	495	92.01
	N = 100	avg = 89.86 %	
Pointing	613	492	80.26
V-Sign	530	475	89.62
Open	594	539	90.74
Closed	587	580	98.81

dilation took 3.8 ms per frame, fingertip detection, gesture recognition and filtering 3.87 ms per frame and Kinect's capture took 22 ms. Regarding reliability, we designed a test session that consisted in performing all the gestures shown figure 5. Each gesture was held for a given number of frames while hand movement was allowed. Table 1 shows a set of results given different filter orders of 24, 42 and 100. The second and third column shows the number of frames the gesture was held, and the number of frames it was positive detected respectively; the fourth column shows detection accuracy given by $(\#detected/\#frames) \times 100$. Notice that for a higher order filter, detection accuracy is better, as expected.

The second experiment was designed to test the feasibility of our solution by performing navigation, selection and manipulation tasks as described previously. In this test we perform empirical observations to verify how well the user interacts with the system. We suggest reviewers watch the companion video which includes a recorded session of this test. As can be seen from the recording, the speed and accuracy of our solution allows real-time interaction in the virtual environments and the user was able to complete successfully each of the proposed tasks.

5 CONCLUSIONS AND FUTURE WORK

On this article, we have described a fast and conceptually simple bare-hand gesture recognition algorithm for real-time interaction in virtual environments. The morphology operations implemented in the GPU and the performed optimizations in fingertip detection stage, consisting in reducing the list of curvature points using equation 2b, result in a low latency hand features detection and tracking stages. The combination of a curvature measure, k-means clustering and the $1 Recognizer with a simple digital integrator give a fast and robust enough recognition system. According to Jakob Nielsen, $0.1s$ is about the limit for having the user feel that the system is reacting instantaneously[1]; our algorithm response time is lower than such limit.

Table 2 summarizes a comparison of our algorithm and other Kinect based methods using similar hardware configurations. Our algorithm's performance is better than current implementations. In ad-

[1]http://www.nngroup.com/articles/response-times-3-important-limits/

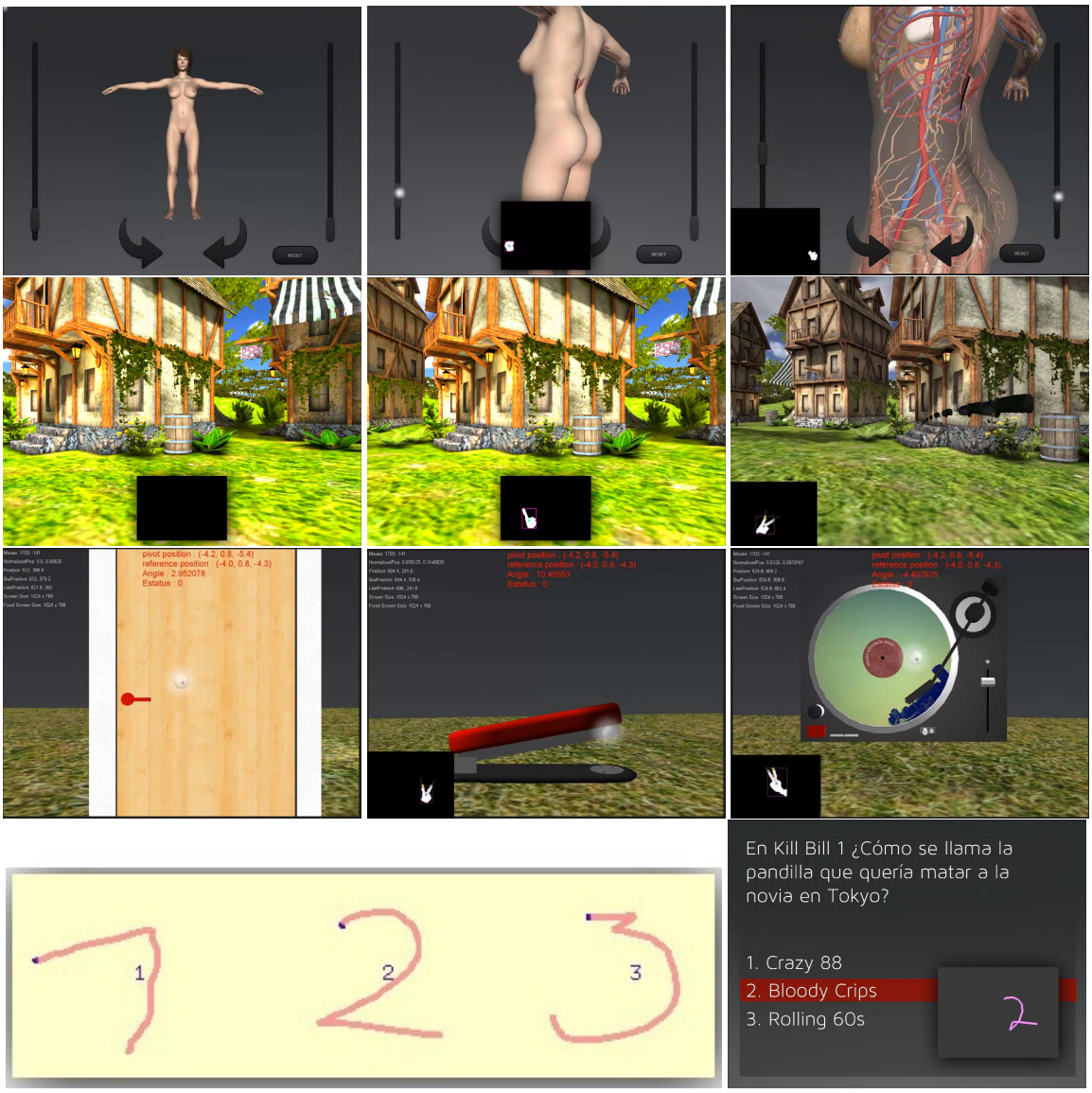


Figure 7: First row: Virtual woman dissection. Second row: Medieval town. Third row: Real life tasks, scratching. Fourth row: Trivia game.

dition, it meets virtual environment requirements, resulting in the seamless rendering of virtual environments.

The technique that is closer in performance than ours, on a similar hardware configuration, is (Suau et al., 2011). However, they downsampled the depth stream data to a resolution of 160x120, which allowed them to recognize only closed and open hand gestures. They also reported a performance of 111 ms for a resolution of 640x480. In (Iason Oikonomidis and Argyros, 2011; Oikonomidis et al., 2011) a robust method for finger tracking is presented, but it does not support gesture recognition. Accurate tracking is achieved using eight cameras turning their solution impractical and expensive. The (Raheja et al., 2011) method is only focused in hand and finger detection and tracking and does not offer specific details regarding performance, testing or applications. Liang et al. (Liang et al., 2012) work is also focused in tracking and their experimental results suggest that its performance is enough for basic virtual environments applications.

On the other hand, the accuracy of our algorithm (86 %) allowed the test subject to successfully complete the tasks as the recorded session demonstrate. The integration of our recognizer into the virtual environments, demonstrated the flexibility of our approach in different interaction situations.

However, despite the fact that using 3D sensing thoroughly simplifies hand detection and tracking, these devices present occlusion problems when fingertips are perpendicular to the sensor. Careful designing of gestures may avoid or reduce this problem. Range distance limits become also a problem when hands are too far from the sensor, i.e. depth information is lost.

Future development and testing is described next:

First, Two hands interaction support. The planned approach is to divide the camera feed into two areas, one for each hand, then perform recognition on these areas. Second, the current implementation uses only the GPU for hand detection and segmentation, performance may improve if all stages are executed on the GPU. Also the implementation of a basic scripting tool to define new gestures will help in order to avoid the modification of the system internals. Finally, The described gestures in section 3.3 were defined for test purposes only, without having usability in mind. However, more gestures have to be specified, and formal evaluation needs to be performed in a group of users in order to know what kind of gestures are usable and suitable for interaction in virtual environments.

REFERENCES

Argyros, A. A. and Lourakis, M. I. A. (2006). Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Proceedings of the 2006 international conference on Computer Vision in Human-Computer Interaction*. Springer-Verlag.

DeFanti, T. and Sandin, D. (1977). Final report to the national endowment of the arts. In *US NEA r60-34-163*.

Frati, V. and Prattichizzo, D. (2011). Using kinect for hand tracking and rendering in wearable haptics. In *World Haptics Conference (WHC), 2011 IEEE*. IEEE Computer Society Press.

Garg, P., Aggarwal, N., and Sofat, S. (2009). Vision based hand gesture recognition. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE Computer Society Press.

Greiner, T. (1991). *Hand Anthropometry of U.S. Army Personnel*. Technical report (U.S. Army Natic Laboratories. Development Engineering Center). U.S. Army Natick Research, Development & Engineering Center.

Iason Oikonomidis, N. K. and Argyros, A. (2011). Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

Iddan, G. J. and Yahav, G. (2001). G.: 3d imaging in the studio (and elsewhere). *In: SPIE*.

Liang, H., Yuan, J., and Thalmann, D. (2012). 3d fingertip and palm tracking in depth image sequences. In *Proceedings of the 20th ACM international conference on Multimedia*. ACM.

Liu, X. and Fujimura, K. (2004). Hand gesture recognition using depth data. In *Proceedings of the Sixth IEEE international conference on Automatic face and gesture recognition*. IEEE Computer Society Press.

Oikonomidis, I., Kyriazis, N., and Argyros, A. A. (2011). Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proceedings of the 2011 International Conference on Computer Vision*. IEEE Computer Society Press.

Plamondon, R. and Srihari, S. N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE Computer Society Press.

Raheja, J., Chaudhary, A., and Singal, K. (2011). Tracking of fingertips and centers of palm using kinect. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2011 Third International Conference on*.

Rehg, J. M. and Kanade, T. (1993). Digiteyes: Vision-based human hand tracking. Technical report, School of Computer Science, Carnegie Mellon University, 1993.

Ren, Z., Yuan, J., and Zhang, Z. (2011). Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia*. ACM.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth movers distance as a metric for image retrieval. In *International Journal of Computer Vision*. Srpinger.

Sturman, D. J. and Zeltzer, D. (1994). A survey of glove-based input. In *IEEE Comput. Graph. Appl.* IEEE Computer Society Press.

Suau, X., Casas, J. R., and Ruiz-Hidalgo, J. (2011). Real-time head and hand tracking. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*. IEEE Computer Society Press.

Utsumi, A. (1997). Direct manipulation scene creation in 3d: estimating hand postures from multiple-camera images. In *Proceedings of ACM SIGGRAPH 97*. ACM.

Wang, R., Paris, S., and Popović, J. (2011). 6d hands: markerless hand-tracking for computer aided design. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM.

Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM.

Zabulis, X., Baltzakis, H., and Argyros, A. A. (2009). Vision-based hand gesture recognition for human computer interaction. In *The Universal Access Handbook*. Lawrence Erlbaum Associates, Inc. (LEA), Series on "Human Factors and Ergonomics".

Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y. (1987). A hand gesture interface device. In *Proceedings of the SIGCHI/GI*. ACM.